Paralelización Parcial del Detector de Bordes de CANNY en CUDA C++

{ Angie Jaramillo, Jhonatan Barrera } Universidad Tecnologica de Pereira

Resumen

En este documento se presenta una comparativa de tiempos de ejecución de la implementación parcial del Detector de Bordes de CANNY en su versión secuencial sobre CPU y dos implementaciones que hacen uso de técnicas de paralelización sobre GPU mediante el uso de CUDA. Se mostrará entonces cómo a partir de estas técnicas de computación en paralelo se obtiene una reducción considerable del tiempo de ejecución dada una aceleración que aumenta a medida de que las dimensiones de la imagen a procesar son mayores.

Trabajo Realizado

El propósito de la detección de bordes en general es reducir significativamente la cantidad de datos en una imagen, siempre y cuando se preserven las propiedades estructurales que se utilizan en el procesamiento de imágenes. Para este fin existen varios algoritmos, y para el propósito de este trabajo estudiaremos el algoritmo desarrollado por John F. Canny (JFC) en 1986 [1].

En este trabajo se implementó parcialmente el algoritmo de Canny en el lenguaje de programación C++, esta implementación la llamamos Secuencial. Se realizaron también dos implementaciones que hacen uso de técnicas de paralelización (sobre CUDA C++), como lo son la paralelización 'base' que hace uso de Memoria Global, y un uso especifico para valores que no cambian a lo largo de la ejecución, Memoria Constante, de manera que se declaran los kernel de Gauss y Sobel como constantes.

El objetivo de estas tres implementaciones es analizar sus tiempos de ejecución y obtener la aceleración resultante de este proceso. Se espera que las implementaciones en paralelo optimicen la ejecución del algoritmo en cuanto a tiempo de ejecución se refiere, mostrando de esta manera que la paralelización es una gran herramienta en la detección de bordes de imágenes complejas al reducir el tiempo de detección de los mismos.

Tiempos

Figura 3: Tiempo de ejecucion para el algoritmo secuencial.

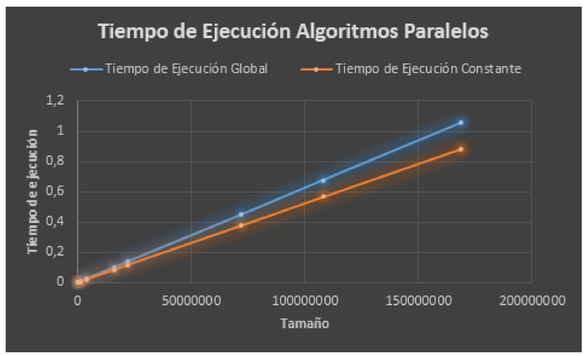

Figura 4: Tiempo de ejecucion algoritmos paralelos.

Tiempos calculados a partir de 20 muestras para 9 imagenes, de tamaños incrementales.

Aceleración

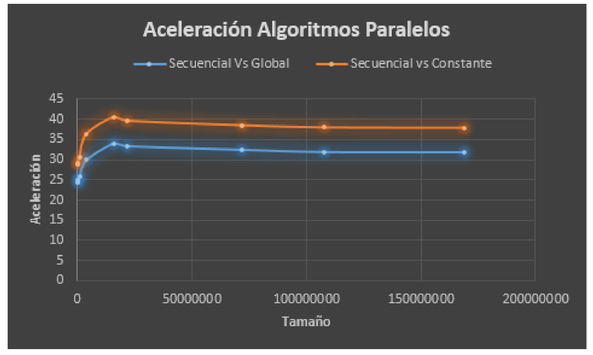

Figura 1: Aceleración obtenida a partir de las implementaciones en CUDA.

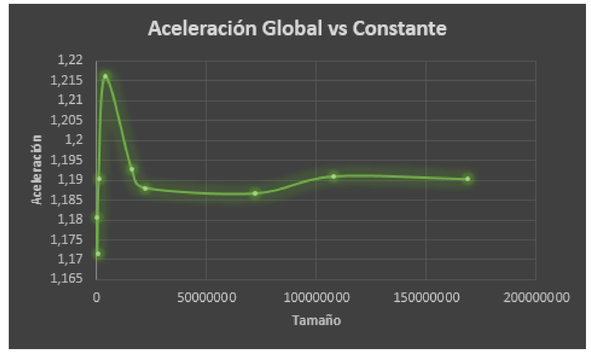

Figura 2: Aceleración entre las dos implementaciones en paralelo.

Como se observa en las graficas 3 y 4, a medida que la matriz de entrada aumenta el tiempo de ejecución tambien lo hace, sin embargo esto apenas y se percibe para las implementaciones en paralelo.

En las Figuras 1 y 2, se ve que la aceleración para las implementaciones en paralelo es notoria, obteniendose una aceleración muy cercana para las versiones en paralelo.

Conclusiones

- Se verificó la factibilidad de paralelizar parcialmente el algoritmo de detección de bordes de Canny, obteniendo resultados favorables en la aceleración del algoritmo.
- Para las dos implementaciones en paralelo que se realizaron se obtuvo una aceleración que permite bajar los tiempos de ejecución entre un $96,9 - 97,4\,\%$ para la prueba con una imagen de matriz 19843x8504.
- La diferencia en la aceleración que se obtuvo entre las versiones paralelas implementadas difiere por poco sin embargo el uso de la memoria constante mejora la implementación sobre memoria global.
- En el análisis de la aceleración obtenida a partir de estas implementaciones paralelas se puede observar que este valor tiende a un valor constate.
- Se debe recordar que la maquina usada para la toma de los datos no es de uso específico y esto puede introducir ruido en los tiempos de ejecución.

Referencias

[1] John Canny. A computational approach to edge detection. Pattern Analysis and Machine Intelligence, IEEE Transactions on , PAMI-8(6):679–698, Nov. 1986

Trabajo Futuro

En este proyecto se paralelizo parte del algoritmo de Canny, queda trabajo por hacer aplicando las técnicas usadas a los pasos de Doble Humbral e Hysteresis.

Se podría mejorar la ejecución de este algoritmo mediante la implementación de otras técnicas de paralelización a estudiar, como el uso de memoria compartida.